EXPRESS MAIL NO. EV 064 092 818 US

Docket No. 01AB162

Q&B Ref: 110003.97745

PATENT APPLICATION FOR

PROCESSOR WITH VERSATILE EXTERNAL MEMORY INTERFACE

by

Joseph Francis Mann

John Robert Figie

PROCESSOR WITH VERSATILE EXTERNAL MEMORY INTERFACE

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] --

STATEMENT REGARDING FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT

[0002] --

BACKGROUND OF THE INVENTION

[0003] The present invention relates to computer architectures adapted for embedded processing and, in particular, to an embedded processor that may work with a variety of different memory types.

[0004] Embedded processors are typically intended to provide control of a specific hardware device into which it is embedded, for example, a household appliance or industrial device. The program executed by the embedded processor is normally stored in a non-volatile memory, for example, flash memory or downloaded from an external source.

[0005] For reasons of cost efficiency, embedded processors normally include the processing unit on the same integrated circuit as other system circuits such as input and output circuits used to communicate the device being controlled and a network interface when the embedded application includes a network connection.

[0006] An exception to this single "chip" integration is the memory system, such as random access memory (RAM) which, for reasons of flexibility, cost, and size may be implemented in one or more separate integrated circuits. Separating the RAM from the embedded processor allows greater flexibility in varying the amount of memory necessary for different embedded applications, allows for more efficient fabrication of the memory, and allows the designer to take advantage of rapidly changing advances in memory technology and design.

[0007] The embedded processor may also incorporate into the integrated circuit a small "boot" memory, typically a read-only-memory holding a "bootstrap program". The term bootstrap refers to the phrase "pulling oneself up by one's bootstraps", that is, raising oneself through one's own, unaided effort. Analogously, the bootstrap program provides the embedded processor with a compact program sufficient so that the embedded processor may load a much more complex program (typically the control program) without external support. The bootstrap program is usually invoked after resetting or at powering-up of the device.

[0008] During the booting process, general memory is required for the temporary storage of register values and other variables. For this reason, the bootstrap program must normally be programmed with sufficient information about the external memory to be able to successfully communicate with this memory. This preprogramming of the boot ROM with the particular external memory setup data limits the use of the embedded processor to that particular memory type.

[0009] What is needed is an embedded processor that may be used flexibly with a wide variety of different external memory types.

SUMMARY OF THE INVENTION

[0010] The present invention provides an embedded processor that may be configured to work with a wide variety of external memory types according to memory setup data read from an external source during the booting process. Prior to having the necessary memory setup data, and thus being able to use the external memory, the bootstrapping program exploits incidental system storage structures such as cache and network buffers to temporarily fill the role of external memory until the external memory has been properly identified and an interface developed.

[0011] Thus, specifically, the present invention provides an integrated processor system comprising an interconnected processing unit performing arithmetic and logical operations, interface circuits for communicating with external devices, and a memory interface for communicating with external memory. A non-volatile boot memory is provided holding a bootstrap program and at least one internal storage structure selected from the group consisting of cache buffers and registers. The processor executes the bootstrap program contained in the non-volatile boot memory using at least one internal storage structure for temporary storage without the use of external memory.

[0012] Thus, it is one object of the invention to provide a processor system using external memory that may execute its bootstrap program before connection with the external memory. In this way, the boot process may identify and configure the memory interfaces for arbitrary external memory types.

[0013] The bootstrap program may provide for the acquisition of external memory setup data required for the communication with external memory.

[0014] It is thus another object of the invention to allow the bootstrap program to use external sources for the memory setup data and thus to allow it to be used with arbitrary memory types simply by changing data in an external source.

[0015] The interface circuits may include a network interface and the bootstrap program may provide for the acquisition of the external memory setup data through a network connection. Alternatively, the external memory may include non-volatile memory and the bootstrap program may provide for the acquisition of the external memory setup data from the external non-volatile memory.

[0016] Thus, it is another object of the invention to provide for a variety of ways of identifying the external memory to the processor system.

[0017] The internal storage structure may be a cache memory and the bootstrap program may execute to read arbitrary data into the cache memory and to lock the cache memory against external reading or writing so that it may be used as variable storage by the processor for further execution of the bootstrap program.

[0018] Thus, it is another object of the invention to use the commonly available cache structure in place of the external memory during the boot process. Prior to connection with the external memory, the cache structure is not required, and can serve in this alternative capacity.

[0019] The processor may require an address translation table mapping internal addresses to physical addresses and, the bootstrap program may make a temporary address translation table in a buffer memory and use the cache memory for temporary storage.

[0020] Thus it is another object of the invention to provide for the generation of a translation table, that may be necessary before use of the cache, through the use of another system storage structure found in buffers often associated, for example, with network interfaces and the like.

[0021] The bootstrap program may further execute after obtaining memory setup data to load additional programs for execution into the random access memory.

[0022] Thus, it is another object of the invention to provide for a normal booting of operational programs into external memory.

[0023] The setup data may be selected from the group consisting of memory type as static or dynamic, memory speed, memory size, memory parity, and memory timing.

[0024] Thus, it is another object of the invention to provide maximum flexibility in linking the processor of the present invention with different types of external memory.

[0025] The foregoing objects and advantages may not apply to all embodiments of the invention and are not intended to define the scope of the invention, for which purpose claims are provided. In the following description, reference is made to the accompanying drawings, which form a part hereof, and in which there is shown by way of illustration, a preferred embodiment of the invention. Such embodiment also does not define the scope of the invention and reference must be made therefore to the claims for this purpose.

BRIEF DESCRIPTION OF THE DRAWINGS

[0026] Fig. 1 is a perspective view in phantom of a display panel for use with an industrial control system such as may include an embedded processor of the present invention and external memory;

[0027] Fig. 2 is a schematic block diagram of the embedded processor and external memory of Fig. 1 showing the integrated processing unit, cache, network interface, boot memory, and memory interface of the processor;

[0028] Fig. 3 is a flow chart showing the steps executed by the processing unit of Fig. 2 in executing the bootstrap program of the boot memory during the boot process;

[0029] Fig. 4 is a diagram of a translation table used by the processing unit in mapping internal memory addresses to memory addresses of an external memory system; and

[0030] Fig. 5 is a detailed flow chart of the execution of the bootstrap program of Fig. 3 in obtaining external memory setup data.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

[0031] Referring now to Fig. 1, an example device 10 incorporating an embedded processor 12 may be a human machine interface (HMI) for an industrial control system or the like. The HMI provides a display panel 15 such as a plasma or liquid crystal display (LCD) presenting information to a user, and a plurality of buttons 18 through which input information may be entered by the user. The buttons 18 and a drive circuit for the display panel 15 may communicate directly with the embedded processor 12 through dedicated input/output lines 21. The embedded processor 12 may also communicate directly with a network 20 and may include a connection 27 to external memory 22.

[0032] Referring now to Fig. 2, in the present example, the embedded processor 12 may include a processing unit 23, including a memory management unit (MMU) 34. The processing unit 23 provides a standard computer architecture allowing for the execution of arithmetic and logical instructions. The MMU 34 provides a mapping of processor internal addresses to addresses of the external memory 22.

[0033] The processing unit 23 connects over an internal bus 31 with a cache 24, network interface circuit 28, input and output (I/O) circuits 30, boot memory 32 and, a memory interface circuit 35. In the preferred embodiment, all of these structures are on a single integrated circuit, however, multiple standardized integrated circuits may also be used and other, additional or fewer structures may also be incorporated into the processing unit 12.

[0034] The cache 24 provides rapid access by the processing unit 23 to the external memory 22 by prefetching address blocks of the external memory 23 using cache protocols well known in the art. Under the guidance of cache control circuitry (not separately shown), a block of data is loaded into the cache where it is read and written to directly by the processing unit 23 to reduce delays inherent in direct communication with the external memory.

[0035] The network interface circuits 28 may include dedicated buffer memory 36 so as to implement a standard network protocol such as EtherNet, ControlNet, DeviceNet, or the like. The I/O circuits 30 may include various I/O registers and necessary buffer and level shifting circuits to provide a communication with the I/O lines 21. I/O circuits 30 may accommodate parallel binary I/O signals or serial signals and analog signals using analog to digital and digital to analog conversion circuitry well known in the art. The boot memory 32 may be a standard read-only memory (ROM) incorporating a bootstrap program as will be described.

[0036] The memory interface circuit 35 provides dynamic and/or static memory control circuits for these different types of external memory. The memory interface circuits 35 include registers (not shown) for holding external memory setup data including, but not limited to, memory type as static or dynamic, memory speed, memory size, memory parity, memory timing. Memory timing may include for example, low level timing specifications for the external memory including, but not limited to, row access strobe (RAS) and column access strobe (CAS) latency. Generally, the memory setup data provides the necessary parameter values for the memory interface 35 necessary to connect to the external memory 22. The memory interface 35 also includes a flag readable by the processor 23 indicating when it has been loaded with external memory setup data 29. This flag need not be in the memory interface as long as it is persistent through a reset.

[0037] Each of these structures is well known in the art and commercially available as standard cells for integrated circuit design.

[0038] The processor unit 23 may connect via the memory interface 35 to the external memory 22 which may include standard random access memory (RAM) 38 and flash memory 40, the former being volatile but the latter being non-volatile and thus suitable for long term storing of additional programs or information. Generally, the flash memory must be of a standard type whose parameters and addresses are contained in the boot memory 30, however, the RAM may be of a wide variety of different memory types and of different sizes. The invention is equally applicable to other memory arrangements as will be appreciated from the following discussion.

[0039] Referring now to Fig. 3, upon boot-up, indicated by process block 42, the embedded processor 12 is directed to a location in the boot memory 32 at the beginning of the bootstrap program. This program then implements a decision block 44 where the processing unit 23 determines whether external memory setup data 29 is present in the memory interface 35 by testing the flag described above.

[0040] Assuming that the external memory setup data 29 has not yet been loaded into the memory interface 35, as will normally be the case upon initial power-up, the program branches to process block 46 where the MMU 34 is initialized by presenting it with a predetermined translation table necessary before use of the cache 24.

[0041] Referring still to Fig. 3, the translation table provides a logical table having columns linked by rows and holding, respectively: an internal address block being part of the address space of the processing unit 23, a flag indicating whether the particular memory block is cacheable with cache 24, and an external address block corresponding in size to the internal address block but mapped to the address space of the external memory 22.

[0042] Following the premise that the embedded processor 12 will work with any kind of external memory 22 or a wide variety of external memory 22, the translation table 48 cannot at this time be accurately completed except for the fixed addresses of the flash memory 40, and thus the predetermined translation table 48 is, in fact, composed in the third column of arbitrarily selected values that may or may not correspond to actual physical memory addresses of the external memory 22 with the exception of at least one row dealing with the flash memory 40. As will be seen, this is of no consequence because external memory will not be used, but the table is prepared only as a prelude to using the cache 24.

[0043] At process block 50, the bootstrap program reads into the cache 24 values from the external memory 22 of desired internal address values, each of which is flagged as cacheable. As the translation table 48 was developed in ignorance of the make-up of the external memory 22, it is likely that these values read into the cache

24 do not have meaningful values. They are loaded into the cache 24 so that, without violating normal cache protocols, subsequent writing and reading of the cache may now occur.

[0044] At process block 52, the cache is locked again following normal cache protocols, preventing a writeback of the cache 24 to the external memory 22. With the locking of the cache 24 at block 52, the bootstrap program may continue execution as indicated by process block 54 to acquire the external memory setup data 29.

[0045] The external memory setup data 29 may come from a variety of sources. Referring now to Fig. 5, at the beginning of process block 54, the bootstrap program determines whether the external memory setup data 29 is contained in flash memory 40 or not as indicated by decision block 66. This may be determined by investigating the known address of flash memory 22 in external memory 22 to see both whether there is any flash memory 40 and if there is flash memory 40, whether the flash memory 40 contains valid external memory setup data 29. As mentioned, the flash memory 40 is of a standard type whose memory setup data is preloaded into the boot memory 30. If flash memory 40 exists, then the program proceeds to decision block 68 where the flash memory 40 is reviewed to determine whether downloading of additional or alternative external memory setup data 29 is allowed. If such downloading is allowed, the program branches to the downloading block 70. If such downloading is not allowed, the program branches to the process block 72 described below.

[0046] Returning to decision block 66, if no flash memory 40 exists, or if the flash memory 40 does not have valid external memory setup data 29, the program proceeds to the downloading block 70 at which the network 20 is interrogated at a predetermined address for external memory setup data 29.

[0047] Assuming that the external memory setup data 29 is obtained either by downloading or from the flash memory 40 (or both) the program proceeds to process block 72 where the external memory setup data 29 is saved in the memory interface 35 and the flag set indicating that valid external memory setup data 29 is contained therein.

[0048] Returning now to Fig. 3, once the external memory setup data 29 is obtained, then at process block 56, the processor is rebooted by jumping to the appropriate location in the boot memory 30 to return to decision block 44. At this point, the memory interface 35 will have been initialized with valid external memory setup data 29 and so, the program branches to process block 58 uses of the external memory setup data 29 to build a full translation table 498 for the MMU in RAM 38.

[0049] At succeeding process block 60, the program determines whether additional programs (e.g. a full operating system or other control programs) to be executed by the processor unit 23 are to be obtained by downloading. If downloading is indicated, the program proceeds to process block 62 to download the necessary programming from the appropriate address. If not, the program proceeds to process block 64 and the program in flash memory 40 as identified by the RAM setup data is executed.

[0050] As will be understood from the foregoing description, the invention employs system storage structures such as buffers, registers, and cache, providing memory like features yet not used during normal operation of the processing unit for general purpose storage, in lieu of standard memory providing general purpose storage during the boot process, thus freeing the designer to use a wide variety of different memory types and sizes in constructing embedded applications.

[0051] It is specifically intended that the present invention not be limited to the embodiments and illustrations contained herein, but that modified forms of those embodiments including portions of the embodiments and combinations of elements of different embodiments also be included as come within the scope of the following claims.